Aalto-yliopisto
Insinööritieteiden
korkeakoulu

Diego Ruiz Ortega

Uncertainty quantification and Sensitivity analysis of the Stirling Engine

Aalto University

Espoo 25.05.2019

Thesis Supervisor: Professor Kevin Otto

Thesis Advisor: Professor Kevin Otto

Aalto University
School of Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi

Abstract of master's thesis

Author Diego Ruiz Ortega

Title of thesis Uncertainty Quantification and Sensitivity Analysis of the Stirling Engine

Master programme Mechanical engineering **Code** ENG25

Thesis supervisor Kevin Otto

Thesis advisor(s) Kevin Otto

Date 27.05.2019 **Number of pages** 64+8 **Language** English

Abstract

The scope of this master thesis is, through different surrogate models, be able to analyse how the different parts of the engine, represented as inputs, interact and affect in the obtainment of the output, represented as power, in a real Stirling Engine. This study used uncertainty quantification and sensitivity analysis.

The thesis structure consists of two major parts.

First major part is a review of the available information that have been used to acquire enough knowledge to be able to complete the thesis.

Second part is an experimental part, python code and matlab code have been used to finally obtain some results that will be analysed in order to see how each input affect to the output power.
This second part is divided into three different chapters.
In the first chapter is going to be explained the python code used for the global sensitivity analysis.
In the second chapter is going to be explained the matlab code used to generate the output power through the different inputs.
In the last chapter, results that have been obtained will be presented and commented.

Keywords Uncertainty quantification, UQ ,Sensitivity analysis, SA , Stirling Engine

Acknowledgements

First, I want to say thank you to Professor Kevin Otto for be my supervisor and help me with everything!

To Aalto University and Finland for let me live an unforgettable year, allowing me to discover new things, new places, new people and also to discover myself a bit better, Kiitos!

Difficult also to forget the people that have help me to feel warmer in this cool place.

Finally, but not less important, REALLY THANK YOU to my father Jose and my mother Concha, without their support most of the things that I have achieve nowadays won't be possible.

Thank you for always being by my side.

"Cuando dejas de aprender, dejas de crecer".

Espoo 27.05.2019

Diego Ruiz Ortega

INDEX

List of Figures

List of Tables

Abbreviations

GPR	Gaussian Process Regression
GSA	Global Sensitivity Analysis
LR	Linear Regression
OAT	One-at-a-time
OED	Optimal experimental design
RBF	Radial Basis Function
RMSE	Root Mean Squared Error
SA	Sensitivity Analysis
SVR	Support Vector Regression
UQ	Uncertainty quantification

1. Introduction

The Stirling engine was invented in 1816 by Robert Stirling[1] and is a thermal system, which is used for the production of power from a high temperature heat source or as a refrigerator and heat pump to deliver energy at a higher temperature than the obtained[2].

Uncertainty Quantification (UQ) is the mathematical field related with the description of uncertainties systems.

It tries to determinate how some outputs can be obtained when the system is not known exactly[3]. In order to achieve the best outcome different methods, need to be selected and combined.

The quality of the UQ analysis carried out depends on the quality of the distributions constructed and the time required to obtain it[4].

The study of how the uncertainty in the output of a model can be apportioned to different sources of uncertainty in the model input is called Sensitivity Analysis (SA)[5].

With SA we can notice which inputs are the contributors and which have no effect[6].

More samples are required in sensitivity analysis than in uncertainty quantification.

In this thesis, I investigate which is the relation between the different parts of the Stirling engine and the generated thermodynamic power variability.

This thesis shows, through different surrogate models, how the different parts of the engine, represented as inputs, interact and affect the performance response, represented as power, in a real Stirling Engine.

This is going to be done using uncertainty quantification (UQ) and sensitivity analysis (SA).

The motivation of this thesis was to know more about a real engine, and in this case the Stirling Engine fitted with that desire.

During the Machine design project course, a Stirling engine was assembly.

As a part of the course students had to measure different parts in order to see the tolerances of each component. After that, we had to manufacture two different components, which were the displacer barrel and the flywheel.

We had additionally to assembly the whole engine, part by part, including our manufactured components, in order to fulfil the requirement and have an acceptable tolerance was quite important.
According to our performance, the speed of the engine would be different.

Since that moment, I was thinking that something related with a Stirling engine could be a good topic, because is a field that I'm interested.

After that, the topic of uncertainty quantification and sensitivity analysis of the Stirling engine fitted with what I was looking for, combining different fields that I'm interested such as statistics, mechanical area, computational software and analysis.

The structure of the thesis consists of two major parts.

First major part is a literature review, included mainly inside each different chapter, where I'm going to explain the different information that during this time, I've been reading in order to acquire enough knowledge to be able to complete the thesis.

Second part is an experimental part, where I've been using python code and matlab code to finally obtain some results that will be analysed in order to see how each input affect to the output power. This second part is divided into three different chapters.

In the first chapter is going to be explained the python code used for the global sensitivity analysis.

In the second chapter is going to be explained the matlab code used to generate the output power through the different inputs.

Finally, in the last chapter, results that have been obtained will be presented and commented.

2. Background

In this part of the thesis, I'm going to explain the different documents and information that I've been looking for with the objective of obtaining enough knowledge to carry out the thesis in a satisfactory way.

Additionally, in each chapter, some literature is going to be used to introduce better the theoretical framework that is related and necessary to later on, have a perfect understanding of the information that will be presented.

The information is mainly related with Stirling Engine[2] because I needed to know how it works and the different variables that affects the output. That was mainly my priority when I looked for information in order to additionally be able to understand the different results that have been obtained with the different tests that have been done.

Also, I have read about Uncertainty quantification (UQ)[6] and Sensitivity analysis (SA)[7] to know better how to focus my thesis.

I needed also literature about python and matlab because some specific information was needed in order to implement really technical scripts.

The investigation about the theory of the thermodynamic analysis of the Stirling engine with sinusoidal and lineal variability in the volume[8] can be used as a starting point of the engine design.
In this scientific paper some important equations are presented.

With this equations[9], some concepts of the engine such as the regenerator dead volume and the different internal heat exchanger are better understood.

The nomenclature used in the present thesis is explained inside, giving some relations between them.

Inside the book Making Stirling Engines[10] we can see a bit of the history of the Stirling engines, including some useful information about the very early design, including some pictures of different examples. Additionally, a lot of information about how works the Stirling engine is provided[10]. This information can be helpful in order to later know better the different results that we will obtain, giving some answer to questions that have come up during the present thesis.

Also, a gamma engine example is provided, with information about it. Our tested engine was also gamma, so the theoretical part about it was useful to understood what was going on in the tests that we did.

In order to know more about the gamma type Stirling engine, information about the influence of the phase angle and dead volume[11] in the engine power has been looked. A CFD simulation has been done in order to see how these parameters are related with the output, obtaining an optimum phase angle and a optimum value for the pipe diameter(dead volume).

Inside SALib[12] we can find Python implementations which are used commonly in the global sensitivity analysis methods, including for example Sobol[13] and Morris[14].

SALib[15] is an open-source of python library, which is useful in the simulation, optimization and modelling systems in order to calculate the impact that can have inputs or some factors in the desired output.

The library provides some functions which will be used to analyse and visualise the results of the model's outputs[16].

A python module called Scikit-learn[17] will be also used to implement the different regression models that will be explained deeper later.

Inside Scikit, we have most of the libraries that will be used in the present document. It is necessary to check the implementation details in order to avoid code issues; some examples will be checked to have a base and a basic knowledge about how to implement the program code[18].

In order to obtain more information about uncertainty quantification, Chen et al. [19] is a nice book due to the fact that he has study automotive engine noise and vibration using GSA. This can be related to the Stirling engine.

It is interesting because additionally information can be extrapolated to other engineering applications where metamodels, based on physical experiments, are constructed.

In order to understand better some concepts such as Morris, Sobol, global sensitivity analysis… Wainwright et. al [20] make a deep explanation about all these concepts that I've been using in this thesis, giving some important information in order to avoid misunderstood.

It provides a comparison of the local sensitivity and two global sensitivity analysis methods such as Saltelli and Morris. Each method will be explained, providing different equations that will simply the understood of some concepts. With the equations [20], we can understand the importance of each parameters, and how it affects to the final output.

As Wainwright et. al [20] talks, in Saltelli method, the number of simulations is higher than in Morris.

Morris doesn't provide the contribution percentage, it just provides indications of which variables are more important and less important. This percentage contribution will be shown in Saltelli.

Additionally, an example is going to be provided, in order to understand even better how it works, explaining the solution that is obtained.

Another example of using Morris method[21] is applied, in this case to know how 28 vegetation parameters affects the grassland system under specific conditions.

3. Experimental Part

This part is divided into three different chapters.

On each part I'm going to do some theoretical explanation about different information related with the topic of the chapter in order to know better about how it works and make understanding easier.

Additionally, the code is going to be explained in order to understand it better.

First chapter is going to be about Global sensitivity analysis, with theoretical explanations in order to understand better the code that is going to be used.

Second chapter is going to be about Stirling engine model, with theoretical explanations and the code that has been used to obtain the output power of each model.

Finally, on the last chapter, called GSA of Stirling engine, we are going to write about the different results that are obtained combining the first chapter and the second, explaining the different result that have been obtained.

4. Global Sensitivity Analysis (GSA)

This first part of the thesis, called Global Sensitivity Analysis, will be done mostly in the programming language called Python.

Global sensitivity analysis[7] will study how the output uncertainty can be divided and allocated into the different inputs[22].

Using this programming language, different scripts will be created and running to obtain different analysis that later the results will be analysed.

Python code library called SALib[15] is going to be used.

In order to be able to use this program, first is necessary to download Anaconda from the official webpage due to Python is inside of it.

On Python, according to which files we are computing, we need different libraries. These libraries can be downloaded from the official webpage[16].

Sobol's method is a form of global sensitivity analysis[13]. It works in a probabilistic framework, decomposing the variance of the output of the model in fractions that are attributed to inputs[23].

Sobol sensitivity analysis[6] determinates the contribution of the different parameter input and their interactions with the total variance of the output model.

Sobol sensitivity analysis intends to determine how much depends on each of the input parameters the variability in model output.

Sobol sensitivity analysis indicates what impact and to what extent will have the input variability on output model[13].

4.1 Sensitivity Analysis

Sensitivity analysis is the study of how the uncertainty in the output of a model can be apportioned to different sources of uncertainty in its inputs.

The sensitivity[15] of each input is represented by the sensitivity index. Sensitivity index come in diverse forms:

1. The first-order index can measure the contribution to the output variance of a single model input.

2. With the second-order index can measure the contribution to the output variance that is caused by the interaction of two model inputs.

3. The total-order index measures the contribution to the output variance of a model input including the first-order effects and all the higher-order interactions.

Sensitivity analysis works on the simple principle of changing the model and check the behaviour according to these changes that are made[24].

The different parameters which we need to observe are:

- The design of the experiment: Includes parameters combination which will vary. Includes a check on how many and which parameters need to be varied at a given point in time, assigning maximum and minimum levels values before the experiment and also study the different correlations.

- What can vary: The different parameters that can be chosen to vary in the model could be some technical parameters, the number of activities or the number of constraints and the limits associated to them.

- What can observe: The value of the different decision variables that we have, the value of a specific objective or the value of the main objective function.

The different steps which are used to bring sensitivity analysis are:

1. Firstly, the base case output needs to be defined; setting an initial input value, for which the sensitivity is going to be measured.

2. Then, the value of the output at a new value of the input, while keeping other inputs constant is calculated.

3. Look for the percentage change in the input and also the percentage that changes the output.

4. The sensitivity is calculated by the division of the percentage change in the output by the percentage change in input.

Finally, the decision maker can have an idea of how sensitive is the solution that is chosen to any variation in the input values of the parameters[25].

4.1.1 Non uniform sampling using distribution functions

Nonuniform sampling is a branch of sampling theory where involved results are related to the Nyquist–Shannon sampling theorem[26].

The interpolation of Lagrange and the relationship between the uniform sampling theorem and itself is the base of Nonuniform sampling[27].

4.1.2 SALib:

SALib[12] is an open source library that is written in Python in order to perform sensitivity analysis.

SALib is the responsible of the generation of the model inputs, using a sample function, and computing the sensitivity indices from the outputs of the model, the analyse function.

The sensitivity analysis using the open source library SALib follows the following steps:

1. First of all, the determination of the inputs of the model that are needed, which are the parameters and their range sample.

2. After, the sample function should be runned in order to do the generation of the model inputs.

3. Evaluation of the model using the inputs that are generated, saving the outputs model.

4. To compute the sensitivity indices, the analyze function on the outputs should be runned.

SALib[12] provides sensitivity analysis methods like Morris, Sobol and FAST[15].

Following we are going to explain deeper Morris and Sobol options.

On each option, I'll explain a little bit of theory about how it works and also the sample and analyze function for each, including the different parameters that are needed.

4.1.3 Morris Sampling

The Morris method[14] for global sensitivity analysis is a called one-step-at-a-time method (OAT), which means that one input parameter has a new value in each run.

It facilitates a global sensitivity analysis by making a number "*r*" of local changes at different points x $(1 \rightarrow r)$ of the possible range of input values[3].

Morris sampling methods allows to classify the inputs in three groups: inputs having large linear effects without interactions, inputs having negligible effects, and also inputs having large non-linear or/and interaction effects.

The method consists in the discretization of the input space for each variable, then performing a given number of OAT design[5].

Inside Morris, we define sample as:

SALib.sample.morris.sample(*problem*, *N*, *num_levels=4*, *optimal_trajectories=None*, *local_optimization=True*)[28].

Model inputs can be generated using Morris Method[14].

It returns a NumPy matrix which contains the model inputs required for Morris Method.

The matrix that results has (G + 1) * T rows and D columns.

In the table 1, we can see the different input parameters that are used in the Morris function that is called sample.

	Name	Class	Explanation
Parameters	Problem	dictionary	The definition of the problem
	N	integer object	The number of trajectories that generates
	num_levels	integer object	The number of grid levels , default value = 4
	optimal_trajectories	integer object	The number of optimal trajectories to sample
	local_optimization	boolean value	Flag whether to use local optimization up the process tremendously for bigger N and num_levels.

Table 1 Input Parameters of Morris sample function

The function return is sample class. It returns a numpy.ndarray, which contains required Method of Morris model inputs.

The resulting matrix has (G / D + 1) * N / T rows and D columns.

The return type is numpy.ndarray.

Inside Morris, we define analyze as:

SALib.analyze.morris.analyze(*problem*, *X*, *Y*, *num_resamples=1000*, *conf_level=0.95*, *print_to_console=False*, *num_levels=4*, *seed=None*)[28].

In the table 2, we can see the different input parameters that are used in the Morris function that is called analyze.

	Name	Class	Explanation
Parameters	Problem	dictionary	The definition of the problem
	X	numpy.matrix	The NumPy matrix contains the model inputs of dtype = float
	Y	numpy.array	The NumPy array contains the model outputs of dtype = float
	num_resamples	integer object	The number of resamples used to compute the confidence intervals
	conf_level	floating point number	Interval level of confidence
	print_to_console	boolean value	Results are printed to console
	num_levels	integer object	Number of grid levels, must be identical to the value passed to SALib.sample.morris

Table 2 Input Parameters of Morris analyze function

The function analyze returns us a dictionary with keys such as “mu”, “mu_star”, “sigma” and “mu_star_conf”[28].

Each entry is a list of parameters which contains the indices in the same order as the parameter file.

Returns:

The sensitivity indices dictionary called **Si** contains the next parameters:

- The mean elementary effect is *mu*
- The absolute of the mean elementary effect is *mu_star*
- Elementary effect of the standard deviation is *sigma*
- The interval confident bootstrapped is *mu_star_conf*
- Parameters names is *names*

The return type for the analyze function of Morris is a dictionary[28].

4.1.4 Sobol Saltelli Sampling

Inside Sobol[6], we define sample as:

SALib.sample.saltelli.sample(*problem*, *N*, *calc_second_order=True*, *seed=None*)[28].

It returns a NumPy matrix which contains the model inputs using Saltelli's sampling scheme.

If the parameter called calc_second_order has False value, the matrix that results has N * (D + 2) rows.

If the parameter called calc_second_order has True value, the matrix that results has N * (2*D + 2) rows.

In the table 3, we can see the different input parameters that are used in the Sobol function that is called sample.

	Name	Class	Explanation
Parameters	Problem	dictionary	The definition of the problem
	N	integer object	Generated number of samples
	calc_second_order	boolean value	Second sensitivities order calculations

Table 3 Input Parameters of Sobol sample function

Inside sobol, we define analyze as:

SALib.analyze.sobol.analyze(*problem*, *Y*, *calc_second_order=True*, *num_resamples=100*, *conf_level=0.95*, *print_to_console=False*, *parallel=False*, *n_processors=None*, *seed=None*)[28].

It returns keys dictionary such as "S1", "S1_conf", "ST" and "ST_conf".

If we have the value of calc_second_order equal to True, then the dictionary also contains "S2" and "S2_conf"[28].

	Name	Class	Explanation
Parameters	Problem	dictionary	The definition of the problem
	Y	numpy.array	A NumPy array containing the model outputs
	calc_second_order	boolean value	Calculate second-order sensitivities
	num_resamples	integer object	The number of resamples
	conf_level	floating point number	Level of confidence interval
	print_to_console	boolean value	Print results directly to console

Table 4 Input Parameters of Sobol analyze function

In the table 4, we can see the different input parameters that are used in the Sobol function that is called analyze.

4.2 Surrogate Models

A surrogate model is a method used to measure an outcome of interest when it can't be measured directly easily.

Nowadays, design problems require simulations or/and experiments to evaluate constraint functions and design objective as a function of variable design. A problem is that most of the times, a simulation can take many hours to be completed[29].

The process comprehends mainly three important steps which may be interleaved iteratively:

- Sample selection, which is also known as sequential design, optimal experimental design (OED) or active learning.
- The construction of the surrogate model and optimizing the model parameters.
- Evaluation of the precision of the surrogate, iterating until the time or design is good enough and fits with our requirements.

The precision of the surrogate depends on the location and also the number of samples. This is related with expensive experiments or simulations in the design space[30].

In our script, before running the regression that we want, we need to run the code block that generates the Saltelli samples, because these are the input samples where we will evaluate the surrogate model at.

That script reads the number of variables, the name of the variables and also our bounds. With this information, according if Saltelli is second order or not, the program generates values in one way or another. After this, the values are interpolated from the input file and later on, with the interpolation done; these values are saved into another csv file which will be used after in the different regression models to compare the results and obtain the correlation.

The following subchapters about the different regression models are implemented in scikit-learn[17].

4.2.1 Linear regression (LR)

Linear regression tries to modelized the interaction between two variables by fitting a linear equation.

One variable is considered to be a dependent variable and the other is considered to be an explanatory variable.

First of all, we need to determine if there is a relation between the interesting variables.

If there are no relation between the dependent variables and the proposed explanatory, then doing a linear regression model will probably not provide a meaningful model.

Correlation coefficient is a number which measure the association between the two variables, which is a value that indicates the strength of the association of the observed data of the two variables[31].

$Y = A + B*X$ is the equation of the linear regression line.

X is the explanatory variable and the term Y is the dependent variable.

A is the intercept value and the slope of the line is the term B[32].

In our script, first we need to load the dataset from our input file. With these values, we get X as data and Y as targets to later get linear regression model with these two variables.

Once this is done, we need to make predictions using the data. We have to check the results and see if are valid, comparing between them.

When this is done, we obtain the root mean squared error (RMSE), and the R squared correlation. In order to make it easier to see the results, we plot the output in a chart of Samples values vs Linear regression Values and we also print the correlation to see how well our prediction model fits.

4.2.2 Support Vector Regression (SVR)

Support Vector Regression (SVR) is a regression method where it tries to fit the error within a certain threshold, which is called boundary limit, and it is represented with the letter Epsilon (ε).

In SVR we need to consider the points which are inside the boundary line that we have defined[33].

In our script, we implemented Support Vector Regression (SVR), using the the scikit-learn[17] so the parameters that we can take belonging to this function are[34] :

C: is the parameter penalty of the error term. Normally the value is 1.

Kernel: It specifies the kernel type to be used in the algorithm. It has to be "lineal", "rbf", "poly", "sigmoid" or "precomputed".

Degree: Polynomial kernel function "poly" degree. This parameter is ignored by all the other kernels.

Gamma: For "rbf", "poly" and "sigmoid" we have this Kernel coefficient. Default value is "auto", which uses 1/n_features.

Coef 0: It is the independent term in kernel function. Only in "poly" and "sigmoid" it is significant.

In the script, first we need to load the dataset from our input file and convert into list. With these values, we get X as data and Y to later get support vector regression with these two variables. Once this is done, we need to make predictions using the data. We have to check the results and see if are valid, comparing between them.

When this is done, we obtain the root mean squared error (RMSE), and the R squared correlation. In order to make it easier to see the results, we plot the output in a chart of Samples values vs SVR values and we also print the correlation to see how well our prediction model fits.

4.2.3 Gaussian Process Regression (GPR)

Gaussian process regression (GPR) is a collection of random indexed variables by space or time, every finite collection of those random variables has a multivariate normal distribution[35].

The distribution of a Gaussian process is the joint distribution of all those random variables. It's a distribution over functions with a continuous domain like for example space or time.

Gaussian process uses a delayed training data until a query is made to the system and a measure of the similarity between points, which is the kernel function, to do a prediction of the value for an unseen point from training data[36].

In our script, inside Gaussian Process Regression (GPR),we can choose different parameters for the regression:

Kernel: The kernel specifying the covariance function of the GP. If none is specified, the kernel "1 * RBF(1.0)" is used as default.

Alpha: It is the noise level in the targets. Larger values correspond to increased level noise level in the observations.

N_restarts_optimizer: The number of restarts of the optimizer for finding kernel's parameters in order to maximize the log-marginal probability. The kernel's initial parameter performed the first run of the optimizer.

In the script, first we need to load the dataset from our input file and convert into list. With these values, we get X as data and Y to later get gaussian process with these two variables.

Once this is done, we need to make predictions using the data. We have to check the results and see if are valid, comparing between them.

When this is done, we obtain the root mean squared error (RMSE), and the R squared correlation. In order to make it easier to see the results, we plot the output in a chart of Samples values vs GP values and we also print the correlation to see how well our prediction model fits.

4.2.4 Radial Basis Function (RBF)

A radial basis function (RBF), is a real valued function, ϕ, which value is only dependent of the distance from the origin, which have the value of $\phi(x) = \phi(\llbracket x \rrbracket)$. It can also be on the distance from some other point called centre, C, which have the value of $\phi(x) = \phi(\llbracket x - c \rrbracket)$.

Any function ϕ which satisfies $\phi(x) = \phi(\llbracket x \rrbracket)$ is a radial function[37].

The sum of radial basis functions are used to approximate given functions[38].

In our script, inside Radial Basis Function (RBF), we can choose different parameters for the regression:

Function: The radial basis function has his base on the radius r, given by the norm, set as Euclidean distance as a default value.

We can choose inside this parameter, different options such as “multiquadric”, which is the default, “inverse”, “gaussian”, “linear”, “cubic”, “quantic” and “thin_plate”.

In the script, first we need to load the dataset from our input file and convert into list. With these values, we get X as data and Y to later get radial basis function with these two variables. Once this is done, we need to make predictions using the data. We have to check the results and see if are valid, comparing between them.

When this is done, we obtain the root mean squared error (RMSE), and the R squared correlation. In order to make it easier to see the results, we plot the output in a chart of Samples values vs RBF values and we also print the correlation to see how well our prediction model fits.

4.3 Tool Change Code in Python, process to obtain the final GSA

Now we are going to explain how, through Jupyter notebook[39], the different parameters that we need to finally have the global sensitivity analysis (GSA) can be obtained.

First of all, we need to import the different libraries that will be used in the future. These libraries are a collection of functions and methods that allows us to perform many actions without writing the code.

After, we need to define the simulation sampling. Inside these script we need to specify where we are going to read the variables, the number of columns in the simulation matrix file, the number of samples for the simulation; the method for simulation sampling that we are going to use (we can choose sobol, halton or latin).

If we choose latin, we need to specify the method (we can use latin hypercube, center, maximin, centermaximin and correlation). Additionally, is necessary to put the number of iterations.
Once all these parameters are filled, we have to put the file name for the simulation input and output samples where the values are going to be saved.

Later on, in the definition for run simulation, is necessary to read the excel file and load the data to lists in order to be able to obtain the variable names, parameters and bound limits.

Now Morris sampling is done in order to know which variables have contribution to the response. Morris doesn't provide the percent contribution, it gives an indicator of which variables are more important, in order to obtain later the Sobol indices calculation with less variables.

According to which method we have choose (sobol, halton or latin), we will compute the input samples and create the number of simulations desired.

It's necessary additionally to convert samples to input space values (between 0,1). We can have normal or uniform distribution.

Once all is done, simulation samples will be written in the desired file that was defined earlier.

Now we need to create the different run folders, where each engine file that is obtained with the matlab code, input and output, will be saved. In order to do it, we create as much folders as input files we have that will be the number of simulations done. When we have put inside

each folder the related input and output, we extract these values and we save them into a csv file that later will be used to compute the different analysis required.

When we have the csv file with all the inputs and the output, we will create the surrogate model from the different options that we have. After that, we have to choose which one we want to be computed. We can run as much surrogate models as we want.

These models are Linear regression (LR), Support vector regression (SVR), Gaussian process regression (GPR) and Radial basis function (RBF).Each surrogate model will give us a different correlation and result.

Finally, we have to choose the surrogate model that we want to generate the global sensitive analysis input.

This analysis will be saved in the excel file with the different information related to it, being able to know the relation of the different variables with the final result and having a plot with all the results.

In the figure 1 the block diagram of the process to obtain the final GSA can be seen.

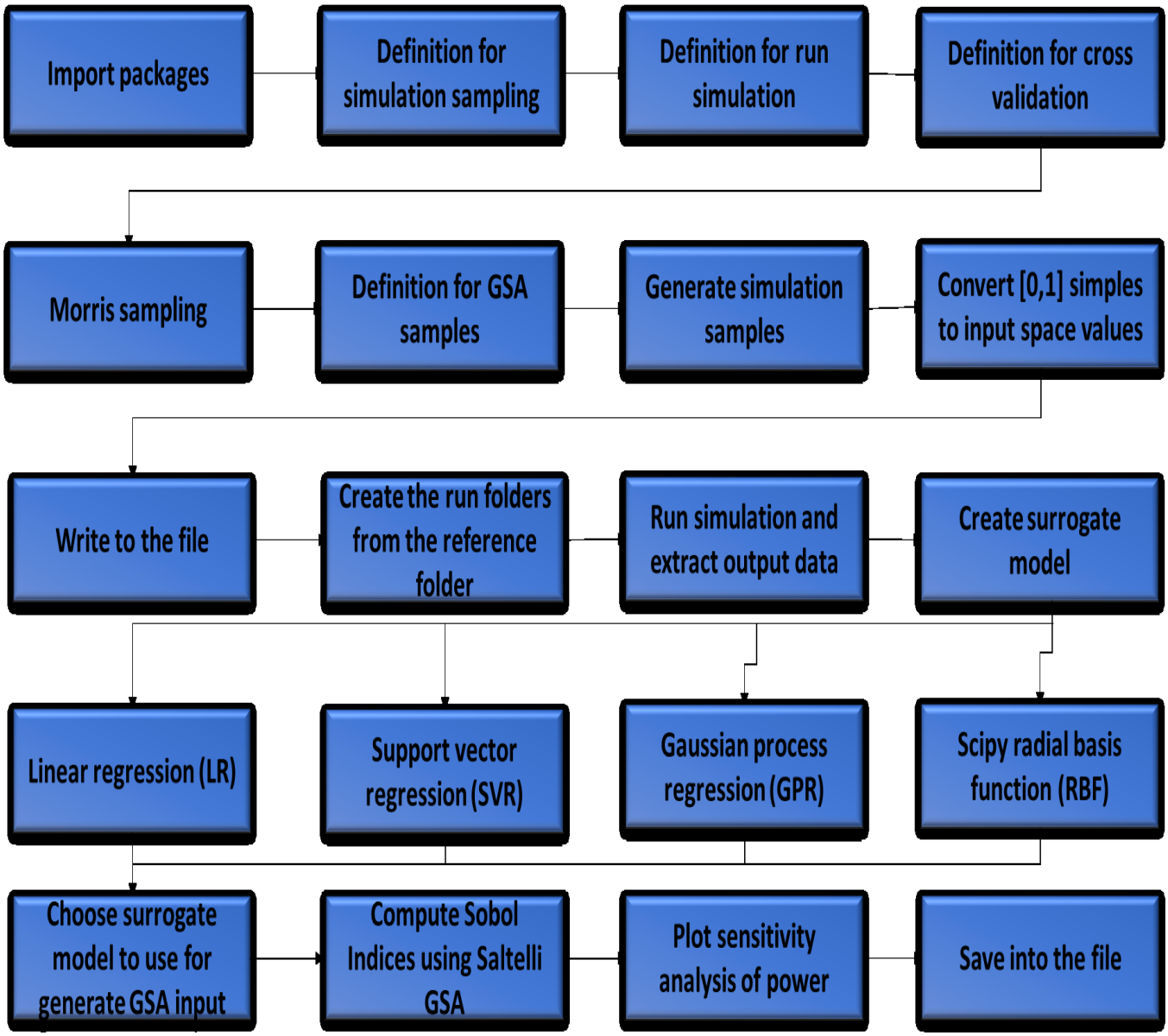

Figure 1 Block diagram to obtain GSA

5. Stirling engine model

A Stirling engine[40] is a heat engine which operates doing a cyclical expansion and compression of air or any other fluid at different temperatures, having a net conversion of heat energy to mechanical work.

Stirling engines can be classified in three types[41].

Alpha Stirling Engine

Alpha Stirling engine contains two separate power pistons in separate cylinders, one cold piston and one hot piston.

The cold piston cylinder is inside the low temperature heat exchanger and the hot piston cylinder is inside the higher temperature heat exchanger.

This type of engine has a very high power-to-volume ratio but has technical problems due to the durability of its seals and also the high temperature of the hot piston.

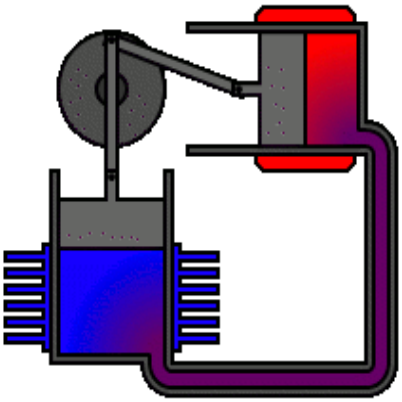

Figure 2 Alpha Stirling Engine

Beta Stirling Engine

Beta Stirling engine has a single power piston arranged inside the same cylinder on the same shaft as a displacer piston.

The displacer piston is a loose fit and doesn't extract any power from the expanding gas, it only serves to shuttle the working gas from the hot heat exchanger to the cold heat exchanger part. When the working gas is pushed to the hot end of the cylinder, it expands and pushes the power piston.

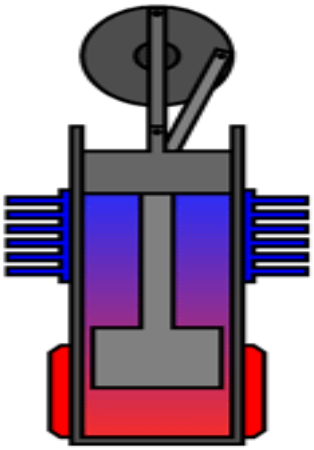

Figure 3 Beta Stirling Engine

When it is pushed to the cold end of the cylinder, it contracts and the

momentum of the machine, usually improved by a flywheel, pushes the power piston the other way in order to do a compression in the gas.

Gamma Stirling Engine

In gamma Stirling engine, the piston of the power is in a separate cylinder alongside the displacer piston cylinder and is attached to the same flywheel.

The gas in the two cylinders can flow with freedom between them and remains a single body.

This configuration produces a lower compression ratio, mechanically is simpler and it is used in multi-cylinder Stirling engines[40].

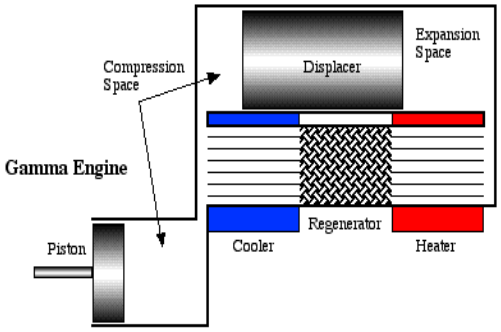

Figure 4 Gamma Stirling Engine

5.1 Matlab Models

Now I'm going to explain the different matlab models Urieli et al. [42] have done and we have been using in order to obtain the different results, that later on will be putted in the python programming code in order to check up on the correlation that we can obtain between the inputs and the output.

5.1.1 Base model of matlab

The ideal Stirling cycle machine can be easily analysed with the use of basic thermodynamics.

However, the analysis of actual Stirling cycle machines has an extremely complexity, because of mainly heat transfer between the external heat sink / source and the working gas, the regenerator, and the non-steady reversing flow of the working gas, requires analysis with sophisticated computer programs[43].

In order to analyse the Stirling engine, we have a program system, created by the university of Ohio, which will be adapted and improved in order to fulfil different requirements.

The code is written in the MATLAB language and includes eight matlab files.

In the figure 5[43] the functional block diagram of the base program can be shown.

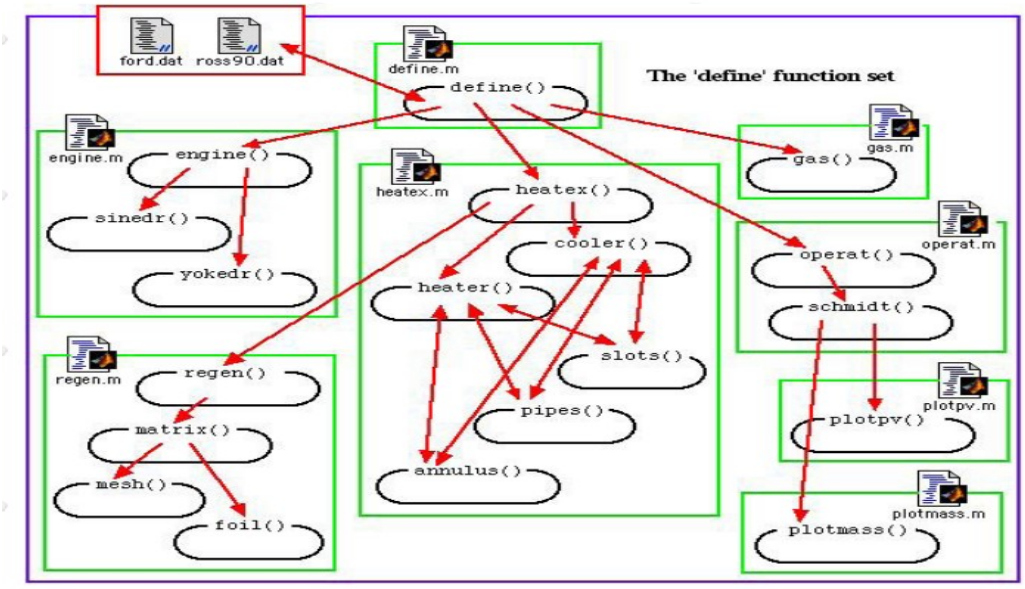

Figure 5 Diagram of the matlab base model[43]

The function set define.m, which is the main function, includes the system definition and the Schmidt analysis.

This function set has the following purposes:

- Inside it, values are specified for the global variables in the define function, needed to define a specific engine configuration. The engine function includes three alpha engines as case studies.

- Specify values for the operating conditions, working gas, and also Schmidt analysis of the engine, in order to be able to evaluate nominal enclosed mass of working gas as well as the Isothermal Schmidt performance of the machine.

- Do a particle trajectory plot using the plotmass.m function

The nineteen functions comprising the global set are included in the eight m-files are: define.m, engine.m, heatex.m, gas.m, regen.m and operat.m; also including two plotting routines which are plotmass.m and plotpv.m.

The global variables required for the simulation are declared in the header of the file called define.m.

The purpose of calling the function set define.m, is the assignation of values to the global variables.

These will be used in the adiabatic.m and simple.m files.

Inside define.m, we generate the different inputs of our program.

Lately we call the functions called engine.m, heatex.m, gas.m and operat.m in order to finally obtain the value of the output, which is the power of our simulated Stirling engine.

Each simulation is a different engine. In our first case, we have a simulation of 1000 iterations, which means a total of 1000 simulated Stirling engines, having each one his own input and output file.

The engine modules inside the matlab script are for Alpha machines, including different options such as Sinusoidal drive, a Ross Yoke-drive and a Rocker-V drive machine.

The heat exchanger types which are included are tubular, annular gap, and slot heat exchangers.

The regenerator matrix types which are included are screen mesh and rolled foil matrices.

The different gas that can be used in the program are air, helium, and hydrogen.

5.1.1.1 Input explanation of the base model

Once the program is runned, we have an input and output for each engine.

Each variable of the input is going to be explained, including the dimension that it has and the matlab file where it is inside.

Inside Engine.m we can obtain the first nine variables which correspond to the rows:

1. Available types of engines (in this case, is beta drive “b”)
2. Piston amplitude (m)
3. Displacer amplitude (m)
4. Displacer phase angle advance (degrees)
5. Piston diameter (m)
6. Displacer diameter (m)
7. Displacer rod diameter (m)
8. Compression space clearance volume (m^3)
9. Expansion space clearance volume (m^3)

The type of engine, which is in the first row, modify the total number of rows. In this case, with a beta drive engine, we have twenty-eight variables.

Inside Heatex.m we can obtain four variables which correspond to the rows:

10. Cooler types (in this case, is smooth pipes “p”)
11. Pipe inside diameter (m)
12. Heat exchanger length (m)
13. Number of pipes in bundle

Inside Regen.m we can obtain six variables which correspond to the rows:

14. Regenerator configurations (in this case, is tubular regenerator “t”)
15. Tube housing external diameter (m)
16. Tube housing internal diameter (m)
17. Regenerator length (m)
18. Number of tubes

19. Matrix type (in this case, is no matrix “n”)

After obtaining the values from the file Regen.m, we need to go back to the file Heatex.m, where we can obtain four variables which correspond to the rows:

20. Heater type (in this case, is smooth pipes “p”)
21. Pipe inside diameter (m)
22. Heat exchanger length (m)
23. Number of pipes in bundle

Inside Gas.m we can obtain one variable which correspond to the row:

24. Type of gas (in this case, is air “ai”)

Inside Operat.m we can obtain four variables which correspond to the rows:

25. Mean pressure (pa)
26. Cold sink temperature (k)
27. Hot source temperature (k)
28. Operating frequency (hertz)

After the different input meanings are explained, in the figure 6 we can see a typical input value file of a Stirling engine with a beta drive type. Each row is the value of a variable.

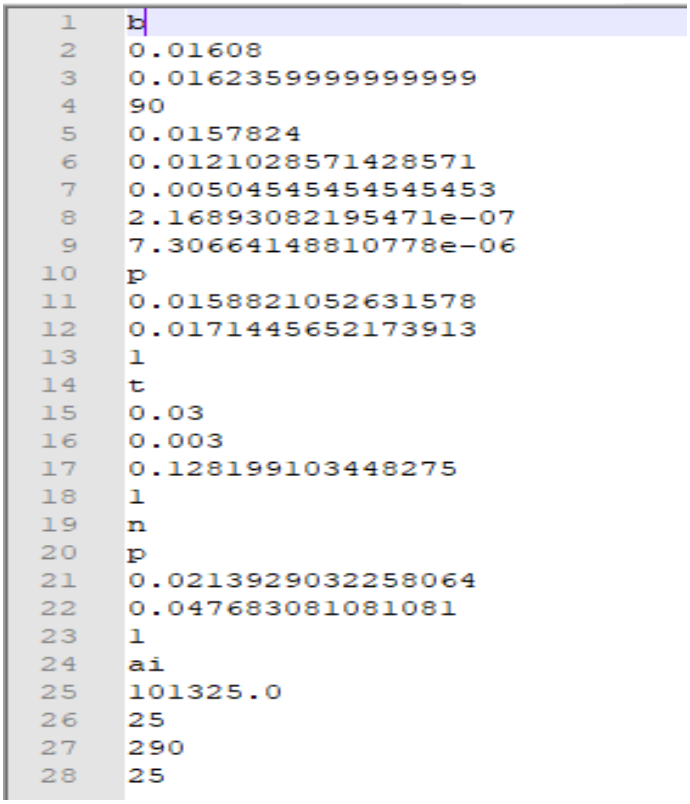

Figure 6 Base model input file

5.1.2 Our Stirling engine model

The original software is created by the Mechanical Engineering Department of Ohio University.

They have created a Stirling Cycle Machine Analysis with different matlab codes which allow us to calculate different options and parameters of a Stirling Engine.

5.1.2.1 Input explanation for our Stirling engine model

In the figure 7 the real sinusoidal type Stirling engine can be seen, where we have measured all the values that later will be introduced in the adaptation version of the base model[43] of the program.

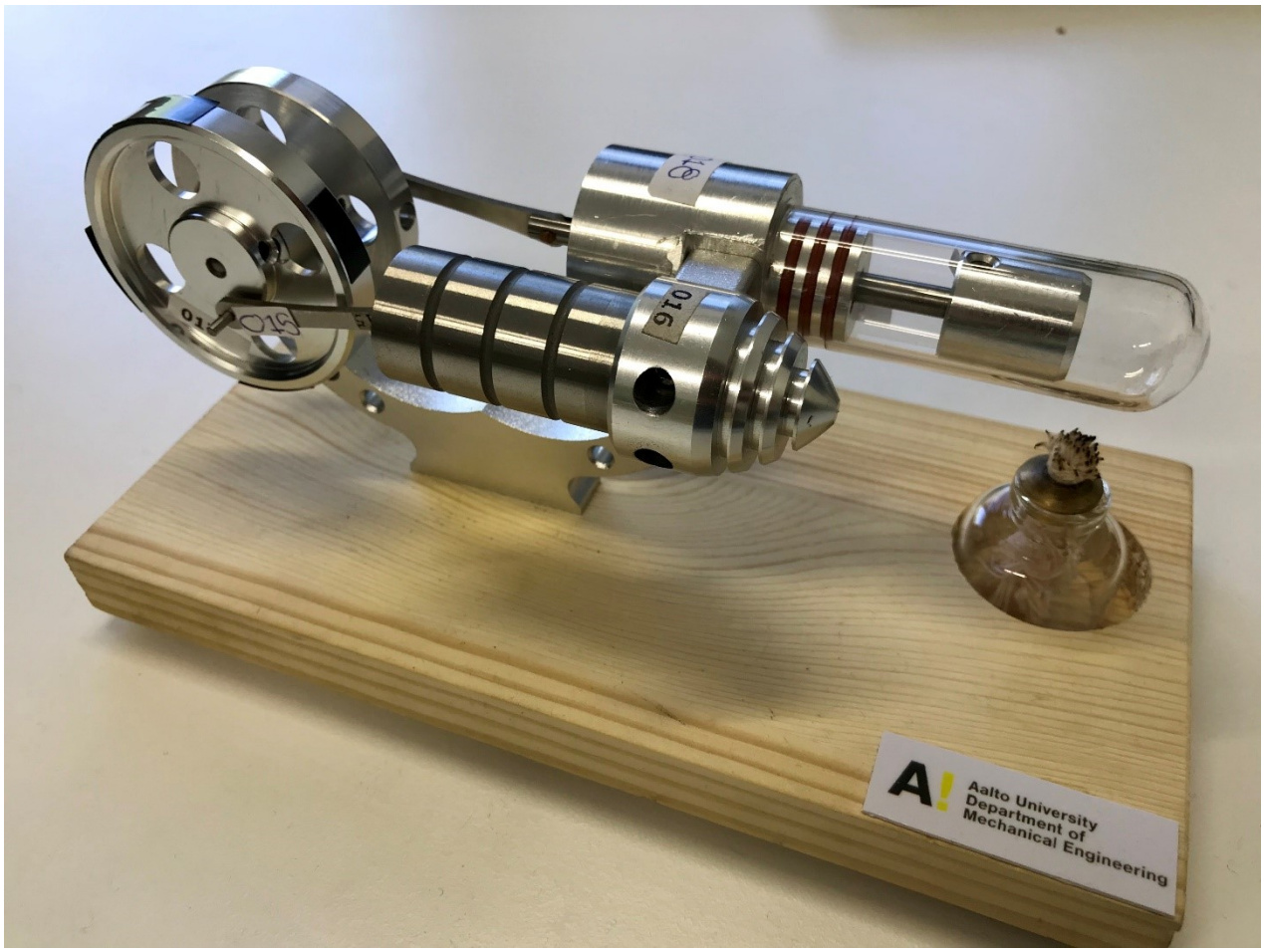

Figure 7 Our Stirling Engine

In order to give the different inputs to the engine model, the twenty five different inputs needed have been measured in the seventeen engines that we have.

Some measures are not directly obtained, we need to compute some calculus in order to have the required value.

In the table 5 and figure 8, we can see the description of the different variables that have been measured in our Stirling engine and are related in all the calculus and also their units.

	Description	Units
dd	Displacer piston diameter	m
dci	Displacer cylinder inner diameter	m
dco	Displacer cylinder outer diameter	m
ldh	The distance between displacer piston(at final position) and the hot side cylinder end	m
Xda	Displacer piston amplitude	m
ld	Displacer piston length	m
ds	Displacer rod diameter	m
Xd	The closest distance the displacer piston travels to the cold side cylinder end	m
lh	Hot side heat exchange length = ldh+Xda	
lr	Regenerator length = ld - Xda	
lc	Cold side heat exchanger length = Xd + Xda	
Rsl	Ring senction length	m
Ldc	Displacer cylinder length	m
Xrd	The distance between the air line connecting the ring section to the displacer barrel	m
dcp	Diameter of the air connector path	m
dh	Diameter of the air line on ring section	
lcd	The air line length connecting the displacer cylinder to the air connector path	
lcd-p	Air connector path length from displacer cylinder to power cylinder	m
lcp	The air line length connecting the air connector path to the power cylinder	m
dpp	Diameter of the air line inide the power cylinder	
Xpp	The vertical distance between the connector air path and power cylinder	m
Xp	The distance between power piston(at final position) and the cold side cylinder end	m
Xpa	Power Piston amplitude	m
lp	Power piston length	m
dp	Powe piston diamerer	m
dpcin	Power cylinder inner diameter	m
Dpc	Power cylnder depth	m
Lpc1	Distance from the end of the power cylinder to the power piston (at initial position)	m
Lpc	Power cylinder length	m

Table 5 Measures explanation

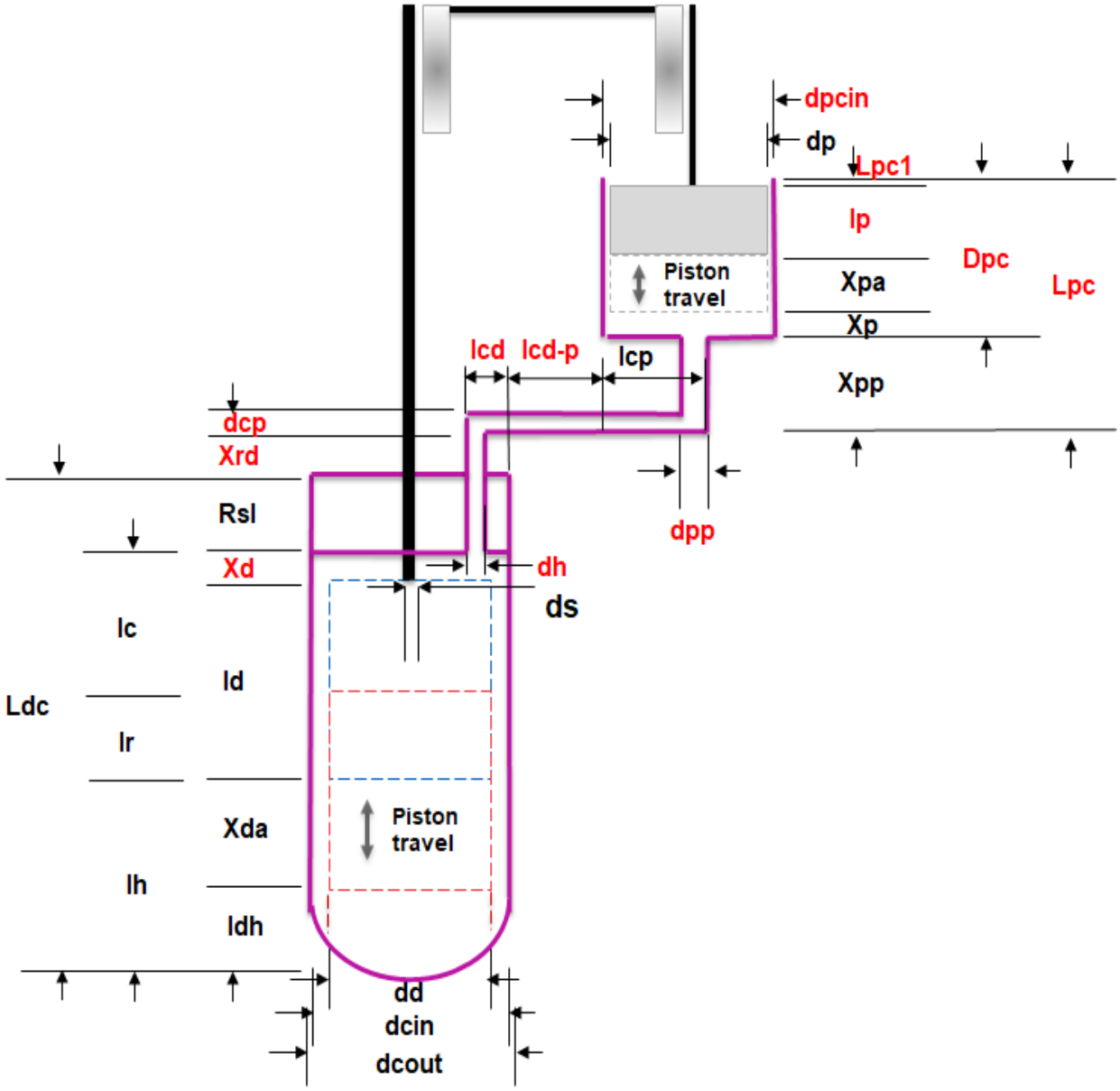

Figure 8 Measures drawing

In the table 6 and figure 9, we can see the different volumes that are related and will be added in order to compute each total volume.

	Description	Units
Vclc	Cold side clearance volumes =(Vclc1+Vclc2+Vclc3+Vclc4+Vclc5)	m^3
Vswc	Compression swept volume =Power piston area * Power piston travel	m^3
Vcle	Hot side clearance volume	m^3
Vswe	Expansion swept volume = Displacer piston area * Displacer piston travel	m^3

Table 6 Volume explanation

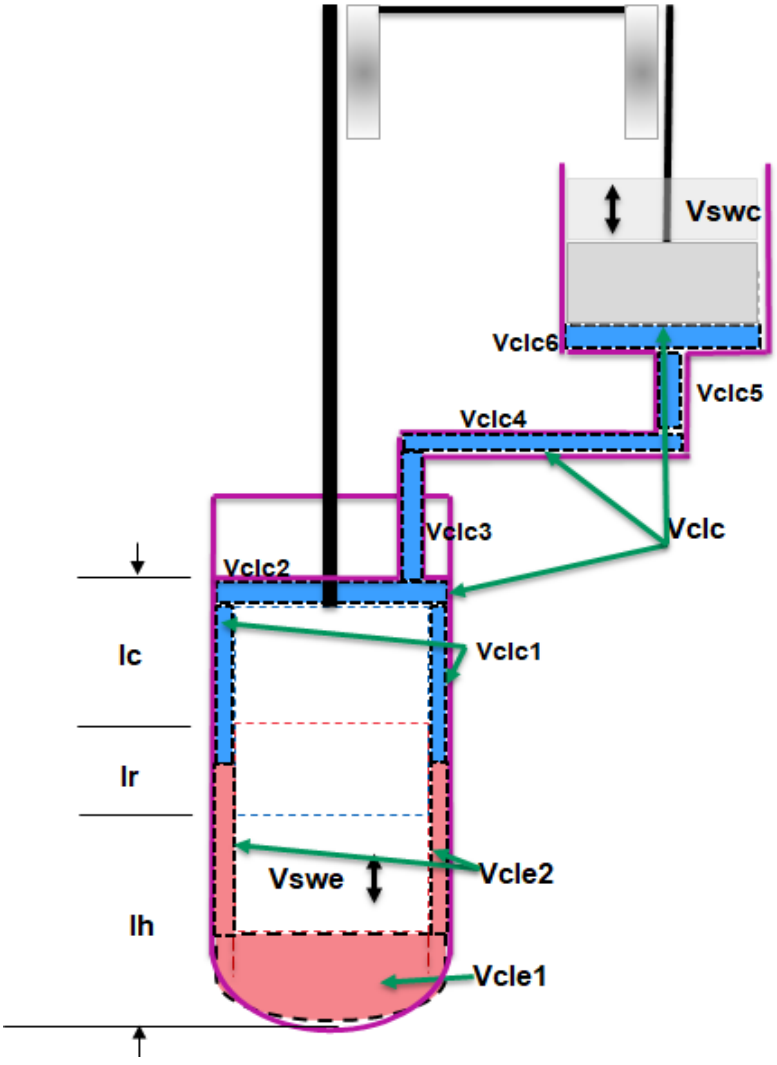

Figure 9 Volume drawing

To do all the measures, we had to disassemble each part of the engine in order to obtain the required measures of each part.

Each measure was done five times by the same person, in order to avoid possible mistakes during the process if we change the person.

After obtaining the five measures, we did the average of the five values to obtain the final result. If the difference between the five values obtained was too high, we repeat the process in order to check that everything was done correctly.

Once this is done, we calculate the value that we need for the input.

After, when we have verified that all values make sense, we have to assembly each part of the engine and make it work properly, testing that everything is fine.

This process should be done with each the engines, a total number of seventeen times.

Finally, when all these input values are obtained, with directly measures or with some calculus, we can see the result in the following page in the table 7.

Engine	Types of engines	Compression space clearance volumes	Compression space swept volume	Expanssion space clearance volume	Expanssion space swept volume	Phase angle	Cooler Type	Annular gap outer diameter	Annular gap inner diameter	Heat exchanger length	Regenerator type	Housing external diameter
0	s	2.43849E-06	3.1769E-06	4.88303E-06	3.84845E-06	90	a	0.019	0.0175	0.0174	a	0.0222
1	s	2.4006E-06	3.23974E-06	4.8517E-06	3.84541E-06	90	a	0.018982	0.017515	0.01736	a	0.022082
2	s	2.60148E-06	3.19732E-06	5.05941E-06	3.86889E-06	90	a	0.01923615	0.0174864	0.01751	a	0.02237615
3	s	2.44919E-06	3.28252E-06	5.00702E-06	3.87162E-06	90	a	0.0191137	0.017498	0.0175	a	0.0222537
4	s	2.45637E-06	3.19644E-06	4.91044E-06	3.86493E-06	90	a	0.019	0.0174992	0.01747	a	0.0222
5	s	2.35929E-06	3.2055E-06	4.70177E-06	3.90832E-06	90	a	0.0188407	0.0175048	0.01764	a	0.0219407
6	s	2.32703E-06	3.25598E-06	4.63791E-06	3.93592E-06	90	a	0.01888852	0.0175234	0.01772	a	0.02204852
7	s	2.35958E-06	3.28428E-06	4.87917E-06	3.86992E-06	90	a	0.01897575	0.0174996	0.01749	a	0.02219575
8	s	2.38946E-06	3.27358E-06	4.7854E-06	3.90422E-06	90	a	0.019	0.017501	0.01763	a	0.02206
9	s	2.38837E-06	3.26507E-06	4.84045E-06	3.91616E-06	90	a	0.019	0.0175008	0.01768	a	0.02208
10	s	2.43553E-06	3.21874E-06	4.88328E-06	3.87945E-06	90	a	0.019	0.0174994	0.01753	a	0.02202
11	s	2.39731E-06	3.2588E-06	4.92249E-06	3.8566E-06	90	a	0.019	0.0174912	0.01745	a	0.02226
12	s	2.48674E-06	3.23683E-06	4.89867E-06	3.93826E-06	90	a	0.019106083	0.0175018	0.01777	a	0.022146083
13	s	2.57148E-06	3.20365E-06	4.96538E-06	3.90859E-06	90	a	0.019205	0.0175108	0.01763	a	0.022445
14	s	2.50436E-06	3.25465E-06	5.03112E-06	3.87349E-06	90	a	0.0191726	0.0174968	0.01751	a	0.0222726
15	s	2.54224E-06	3.25891E-06	5.11403E-06	3.88849E-06	90	a	0.01926825	0.0175252	0.01752	a	0.02232825
16	s	2.59721E-06	3.238E-06	5.14312E-06	3.90662E-06	90	a	0.01930575	0.017501	0.01764	a	0.02244575
17	s	2.70942E-06	3.17571E-06	5.24928E-06	3.88943E-06	90	a	0.01938875	0.0175002	0.01757	a	0.02250875

Housing internal diameter	Matrix internal diameter	Regenerator length	Matrix type	Heater Type	Annular gap outer diameter	Annular gap inner diameter	Heat exchanger length	type of gas	mean pressure (pa)	cold sink temperature	hot source temperature	operating frequency
0.019	0.0175	0.0039	n	a	0.019	0.0175	0.0335	air	101325	25	290	10
0.0168982	0.0157515	0.00394	n	a	0.0168982	0.0157515	0.0330444	air	101325	25	290	10
0.0192363615	0.0174864	0.0038118	n	a	0.0192363615	0.0174864	0.0330439	air	101325	25	290	10
0.0191137	0.017498	0.0037465	n	a	0.0191137	0.017498	0.0330454	air	101325	25	290	10
0.019	0.0174992	0.003883	n	a	0.019	0.0174992	0.0330466	air	101325	25	290	10
0.0188407	0.0175048	0.0037104	n	a	0.0188407	0.0175048	0.0330463	air	101325	25	290	10
0.0188808852	0.0175234	0.0035564	n	a	0.0188808852	0.0175234	0.0330335	air	101325	25	290	10
0.0189787575	0.0174996	0.0038336	n	a	0.0189787575	0.0174996	0.0330365	air	101325	25	290	10
0.019	0.017501	0.003867	n	a	0.019	0.017501	0.0330377	air	101325	25	290	10
0.019	0.0175008	0.003862	n	a	0.019	0.0175008	0.0330361	air	101325	25	290	10
0.019	0.0174994	0.003877	n	a	0.019	0.0174994	0.0330362	air	101325	25	290	10
0.019	0.0174912	0.0038851	n	a	0.019	0.0174912	0.0330367	air	101325	25	290	10
0.0191068083	0.0175018	0.003512	n	a	0.0191068083	0.0175018	0.0330354	air	101325	25	290	10
0.019205	0.0175108	0.003872	n	a	0.019205	0.0175108	0.0330322	air	101325	25	290	10
0.0191726	0.0174868	0.003875	n	a	0.0191726	0.0174968	0.0330353	air	101325	25	290	10
0.0192682825	0.0175252	0.0037485	n	a	0.0192682825	0.0175252	0.0330355	air	101325	25	290	10
0.0193930575	0.017501	0.003865	n	a	0.0193930575	0.017501	0.0330359	air	101325	25	290	10
0.0193938875	0.0175002	0.0038696	n	a	0.0193938875	0.0175002	0.0330361	air	101325	25	290	10

Table 7 Input matrix matlab

The input values that will be introduced to the matlab program will have the following format, which is shown in the figure 10.

```
s    sinusoidal
=vclc   Cold side clearance volumes
=vswc   Power piston area * power piston travel
=vcle   Hot side clearnce volume
=vswe   Displacer piston area * displacer piston travel
90  phase angle
a   cooler
=dout   displacer cylinder inner diameter
=din    displacer piston diameter
=lc    cold side length
a   regenerator
=dout    displacer cylinder outer diameter
=domat   displacer cylinder inner diameter
=dimat   displacer piston diameter
=lr      regenreator length = displacer piston length - displacer pistron travel
n   no matrix
a   heater
=dout   displacer cylinder inner diameter
=din    displacer piston diameter
=lh     hot side legth
ai   air
101325.0  pressure
25   Cold side temperature
290  Hot side temperature
10   Rotating frequency Hz
```

Figure 10 Input values matlab

The table 8 explains deeper the different inputs, saying the meaning, the value that it could have, the units of the dimension, the nomenclature and the meaning of that nomenclature.

We can split the inputs in different part of the engine.

The inputs from seven to ten are related with the cooler part.

The inputs from eleven to sixteen are related with the regenerator part.

The inputs from seventeen to twenty are related with the heater part.

	Inputs	Engine	Value	Units	Nomenclature	Nomenclature meaning
	1	Types of engines	s			Sinusoidal
	2	Compression space clearance volumes	According to engine geometry	m^3	vclc	Cold side clearance volumes
	3	Compression space swept volume	According to engine geometry	m^3	vswc	Power piston area * power piston travel
	4	Expanssion space clearance volume	According to engine geometry	m^3	vcle	Hot side clearance volume
	5	Expanssion space swept volume	According to engine geometry	m^3	vswe	Displacer piston area * displacer piston travel
	6	Phase angle	90	degrees		
Cooler	7	Cooler Type	a			Annulus
Cooler	8	Annular gap outer diameter	According to engine geometry	m	dgap	Displacer cylinder inner diameter
Cooler	9	Annular gap inner diameter	According to engine geometry	m	din	Displacer piston diameter
Cooler	10	Heat exchanger length	According to engine geometry	m	lc	Cold side length
Regenerator	11	Regenerator type	a	m		Annulus
Regenerator	12	Housing external diameter	According to engine geometry	m	dout	Displacer cylinder outer diameter
Regenerator	13	Housing internal diameter	According to engine geometry	m	domat	Displacer cylinder inner diameter (dgap)
Regenerator	14	Matrix internal diameter	According to engine geometry	m	dimat	Displacer piston diameter
Regenerator	15	Regenerator length	According to engine geometry	m	rl	Regenreator length = displacer piston length - displacer pistron travel
Regenerator	16	Matrix type	n			No Matrix
Heater	17	Heater Type	a			Annulus
Heater	18	Annular gap outer diameter	According to engine geometry	m	dgap	Displacer cylinder inner diameter
Heater	19	Annular gap inner diameter	According to engine geometry	m	din	Displacer piston diameter
Heater	20	Heat exchanger length	According to engine geometry	m	lh	Hot side length
	21	type of gas	"ai"			Air
	22	mean pressure (pa)	101325	pa		
	23	cold sink temperature	25	°C		
	24	hot source temperature	290	°C		
	25	operating frequency	10	Hertz		Rotating frequency Hz

Table 8 Explanation input matlab

5.1.2.2 Matlab code for our Stirling engine model

Once we have all these values, I have created the following code in order to read all these input values and be able to calculate the output power.

This Matlab code works with sinusoidal Stirling engines.

The code that is in the appendix 1: Matlab code, reads the 17 rows of the excel file (each row is an engine).

On each engine, we have 25 columns with the different parameters that we have to read.

We save each column in a different variable in order to compute later the power.

The variables are defined as global to make changes to the said variable in different functions if needed.

Later, when the different variables are loaded, using different equations we can compute them to obtain the power.

Power variable is created as long and also will be saved with 15 decimals in order to don't be round off.

Once we have all the power values, we will save in an excel file with the 17 rows and 1 column.

If we want to include more engines, we only need to modify the values of the variable "i" in order to adapt it to our new case of engines.

As we can see in the table 9, the speed of engines evaluated varied from 300 to 450 rpm. The histogram of the speed variation is shown in the figure 11.

Number of engine	Speed (rpm)
1	390.483871
2	394.3541935
3	376.935
4	376.9354839
5	385.9615385
6	366.7741935
7	373.7903226
8	385.4032258
9	374.7580645
10	365.0806452
11	398.9516129
12	336.5322581
13	404.516129
14	403.3064516
15	343.3064516
16	345.7258065
17	314.2741935

Table 9 Speed of each engine

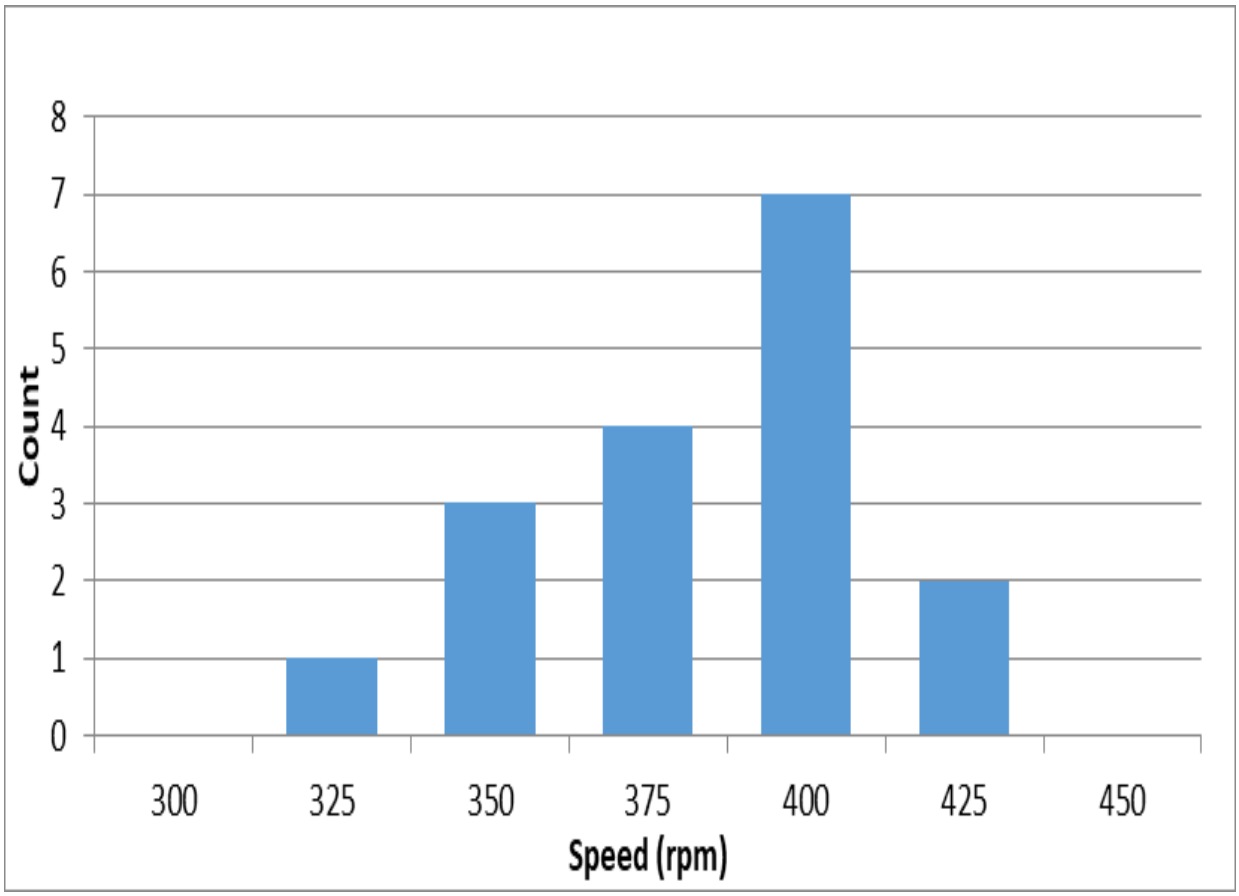

Figure 11 Histogram of Stirling engine Speed

This gap in the speed is due to the variability in the fabrication and also in the assembly of the engine. Both parameters made the difference in the engine performance obtained.

In order to investigate about the causes of the power variability, the steps shown in the block diagram of the figure 1, were done.

The power value predicted in the 1000 UQ runs can be seen in the Table 10. The histogram of the figure 12 represents the distribution of the power. The values of power vary mainly from 1.61 to 1.71 Watts.

Power (W)	Count
1.53	0
1.55	1
1.57	7
1.59	44
1.61	88
1.63	180
1.65	216
1.67	187
1.69	160
1.71	77
1.73	26
1.75	13
1.77	1
1.79	0

Table 10 Power value predicted

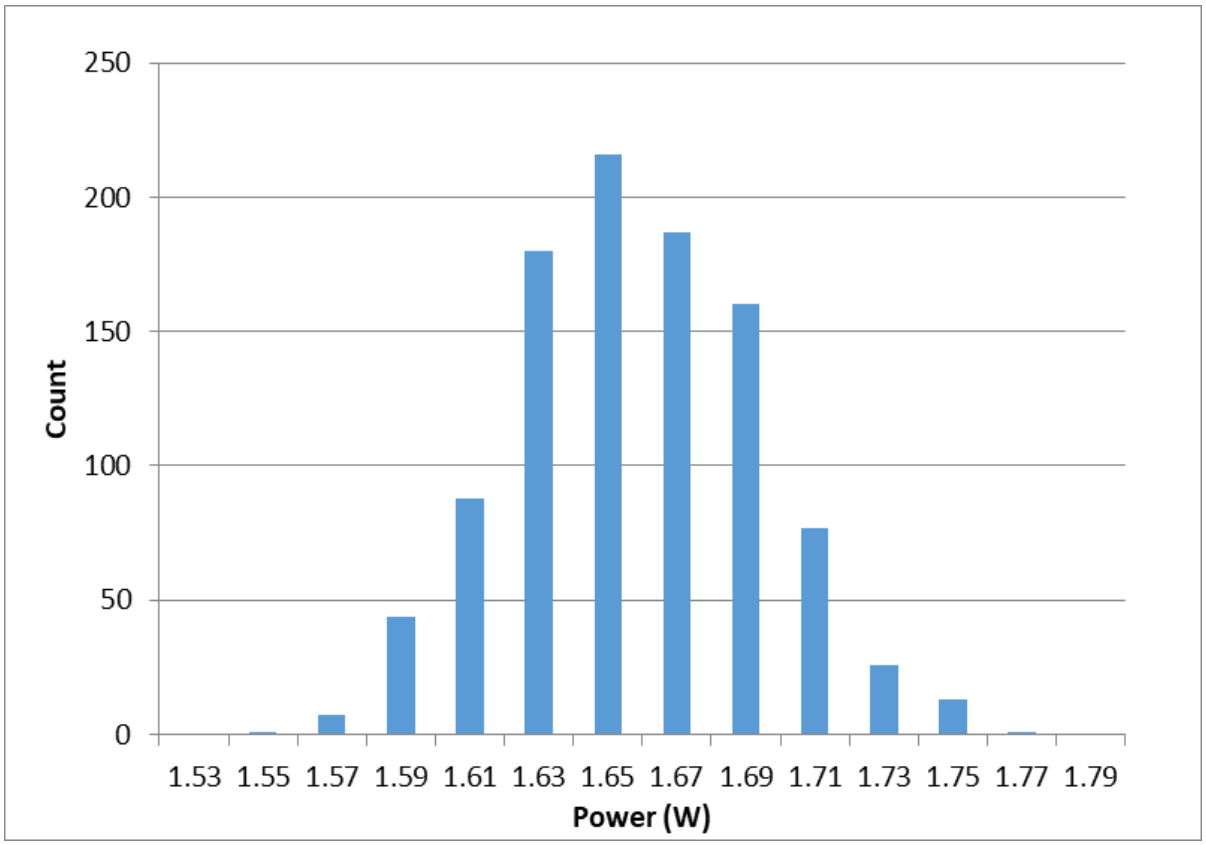

Figure 12 Histogram of Stirling engine Power

A conclusion about the data which have been obtained, will be presented in the next chapter.

5.1.3 Engines Morris samples

This code is an adaptation of the base model of matlab[42], modifying different parameters and values in order to be able to obtain the results that we want for the Morris samples[14].

In this case, I'm going to read a csv file which contains 16000 rows. With the matlab script and these input values, I'll obtain the output power.

The matlab program is going to read the input files from the csv and is going to save the result in the same csv file, in an additional column at the end called power. In order to keep the names of the first row, which are the name of the variables, I have created a cell instead of a matrix like in the previous matlab scripts because in a matrix is not possible to keep the name in the first row of the file.

On each engine, we have 15 columns with the different parameters that we have to read.

We save each column in a different variable in order to compute later the power.

The variables are defined as global to make changes to the said variable in different functions if needed.

Later, when the different variables are loaded, using different equations we can compute them to obtain the output power.

At the end of the script, once we have all the power values, we will save in the same file the 16000 rows in an additional column at the end.

In case that we want to include a different number of engines, we only need to modify the values of the variable "i" in order to adapt it to our new case of engines, putting the desire value to calculate.

5.1.4 Engines Saltelli samples

This code is an adaptation of the base model matlab[42], modifying different parameters and values in order to be able to obtain the results that we want for the 17000 Saltelli samples off the 1000 base samples used in the UQ.

In this case, I'm going to read a csv file which contains 17000 rows, which means 17000 engine inputs. With the matlab script and these input values, I'll obtain the output power.

The matlab program is going to read the input files from the csv and is going to save the result in a excel file.

On each engine, we have 15 columns with the different parameters that we have to read.

We save each column in a different variable in order to compute later the power.

The variables are defined as global to make changes to the said variable in different functions if needed.

Later, when the different variables are loaded, using different equations we can compute them to obtain the output power.

At the end of the script, once we have all the power values, we will save in an excel file with the 17000 rows and 1 column.

If we want to include a different number of engines, we only need to modify the values of the variable "i" in order to adapt it to our new case of engines.

6. GSA of Stirling Engine

In this chapter I'm going to write about the different results that have been obtained combining the previous two chapters, Global sensitivity analysis and Stirling engine models.

6.1 Verification of the Surrogate Models

First of all, I have to verify that the different surrogate models work properly and the results that are obtained are valid.

I have to check the four different surrogate models that we have, which are Linear regression (LR), Support vector regression (SVR), Gaussian process regression (GPR) and Radial basis function (RBF) and make them fit properly with different inputs and be able to plot the results with a right result.

In this case, it is going to be explained how to verify the Radial basis function (RBF).

In order to check and verify that our surrogate model work properly with a good result, I'm going to create a vector which contains the values of a csv file. The first 50% of the rows check how the surrogate model works and later predict the last 50% of the rows, in order to compare the results.

This is done with a python script, where I read the values from the csv file, to save it in a vector with the length of the rows / 2. Later a surrogate model function will be created. This function reads the input values and response values and evaluates them.

After that, we compute the equation using the first set of values. With this function, we are going to compute the second set of values. With this second set of values, we can compare them with the known responses.

Now we can graph the Surrogate model values vs power values, printing also the root mean square error (RMSE) and R-squared.

This information can be seen in the figure 13.

```
RMSE:  0.023
Rsq:  0.8993610875408352
```

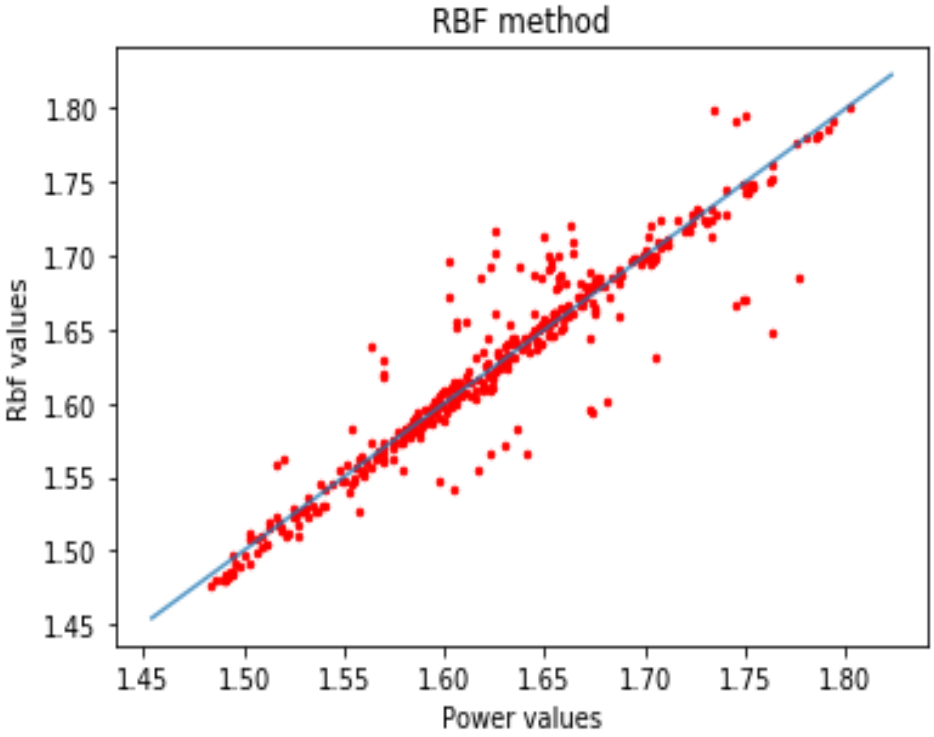

Figure 13 RBF method verification plot

Additionally, another verification that I've done in order to check that the different surrogate models work properly, is modify different parameters to see how it adapts.

In this case, with random values, the surrogates models don't fit properly because there is not relation between them, which is a good and valid result.

Modifiying slightly some values, the root mean square error and the r-squared changes. If we put values that are far from the rest, the results get worse, which is logical, and if we modify some far values putting them close to the others, the results get better, which fits with the expectation and allow us to check that everything is done correctly.

6.2 GSA Analysis done

Here I'm going to explain the different analysis that have been done and the results that are obtained.

First, we have to do Morris in order to obtain the provides indications of which variables are more important and less important.

After, with Saltelli, the percentage contribution will be calculated.

6.2.1 Morris Analysis

Now the results that we have obtained in the Morris analysis[14] will be explained.

The measure Sigma, is used to quantify the amount of variation of a data values group[44].

Morris indices called mu (μ) and sigma (σ), are computed for the input variables, including their confidence intervals.

The inputs that have high values of μ, contribute as a main effect; with high values of σ, contribute as order effects.

Those values with small μ and σ, don't contribute to the variability of the output, which in this case is the power.

As we can see in the figure 14, according to the results, we can be noticed that several variables do not contribute too much, such as matrix internal diameter and phase angle.

The contribution variables are mainly displacer cylinder diameter, cold side clearance volume and also power piston swept volume.

In the figure 15 we can see a plot where we compare each variable Sigma vs Mu*, to see the results that were presented earlier.

Morris doesn't provide the contribution percentage, it just provides indications of which variables are more important and less important. This percentage contribution will be shown in the following Saltelli subchapter.

Variables	mu	conf_mu	mu_conf	sigma	Meaning
dhout	0.165108	0.213265	[-0.048, 0.378]	0.303892	Housing external diameter
dhin	0.254406	0.163358	[0.091, 0.418]	0.326939	Housing internal diameter
Vswe	0.311238	0.262487	[0.049, 0.574]	0.341639	Displacer piston swept volume
gapin	0.119438	0.119855	[-0.0, 0.239]	0.199797	Annular gap inner diameter
gapout	0.24383	0.262228	[-0.018, 0.506]	0.416187	Annular gap outer diameter
Vswc	0.344635	0.349004	[-0.004, 0.694]	0.463251	Power piston swept volume
domat	0.208027	0.119326	[0.089, 0.327]	0.246173	Annular gap inner diameter
lc	0.134944	0.10898	[0.026, 0.244]	0.194886	Heat exchanger length
Phikno	0.075192	0.068888	[0.006, 0.144]	0.110389	Phase angle
dimat	0.062905	0.042622	[0.02, 0.106]	0.058732	Matrix internal diameter
dout	0.383414	0.292851	[0.091, 0.676]	0.534076	Displacer Cylinder diameter
rl	0.138987	0.106185	[0.033, 0.245]	0.164741	Regenerator length
Vclc	0.350585	0.366371	[-0.016, 0.717]	0.470989	Cold side clearance volumes
Vcle	0.219143	0.325651	[-0.107, 0.545]	0.448958	Hot side clearance volume
lh	0.182133	0.246445	[-0.064, 0.429]	0.321036	Heat exchanger length

Figure 14 GSA Analysis Morris

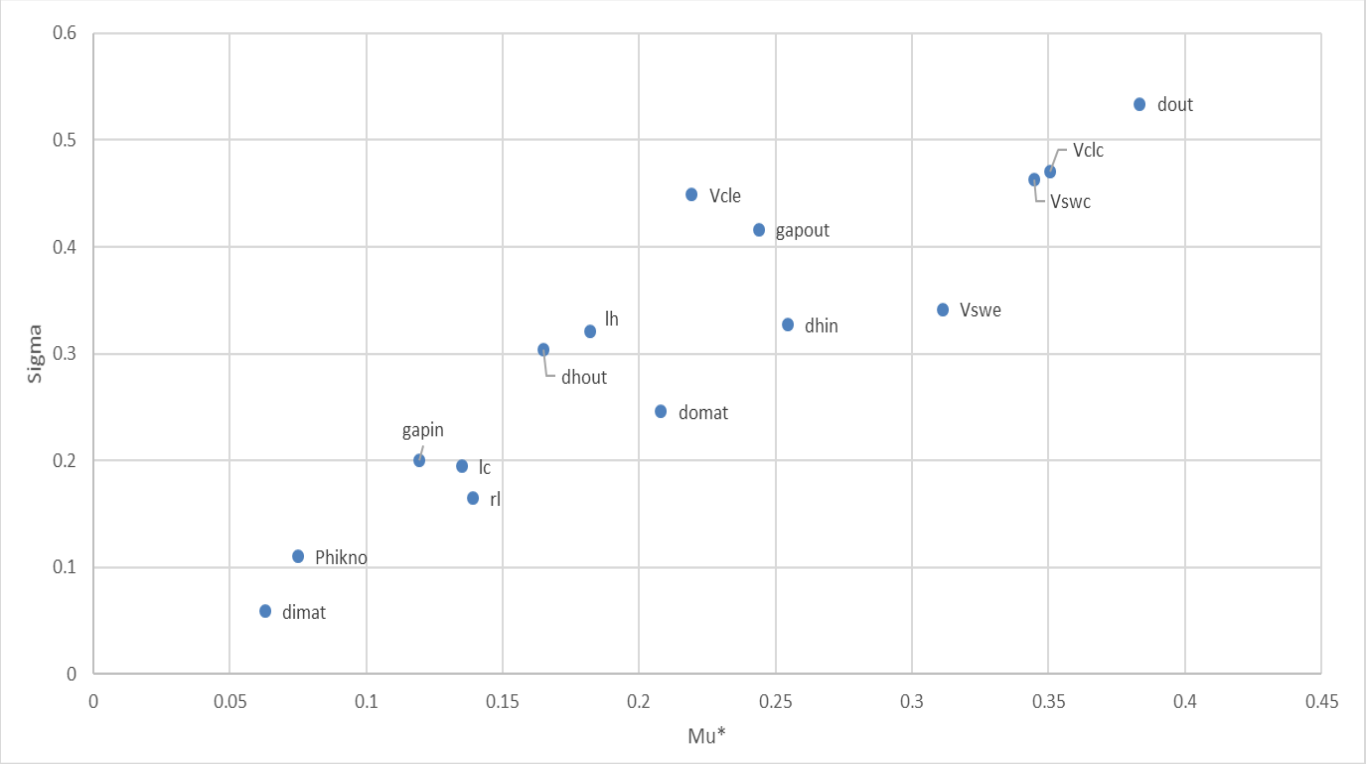

*Figure 15 Morris plot Sigma vs Mu**

6.2.2 Saltelli Analysis

Now I'm going to explain the results that we have obtained in the Saltelli analysis[6].

S1 is the input variable main effect sobol index. It shows the variance of the performance response due to individual variable alone.

The inputs with high S1 contributes as a main effect.

ST is the total effect sobol index. Small values of ST do not contribute to the variability of the response.

The inputs with high ST contribute as higher order terms.

Saltelli calculates the percentage contribution of each variable.

As we can see in the figure 16, according to the results, the variables that have more influence in the output power are the annular gap outer diameter and the compression space clearance volume.

Additionally, we can see in the figure 16, we can be noticed that several variables do not contribute too much, such as matrix internal diameter and phase angle.

Variables	S1	Conf_S1	S1_Conf	ST	Conf_ST	ST_Conf	Meaning
Vclc	0.316486	0.04076	[0.276, 0.357]	0.318413	0.025117	[0.293, 0.344]	Cold side clearance volume
Vswc	0.043568	0.017993	[0.026, 0.062]	0.039448	0.003578	[0.036, 0.043]	Power piston swept volume
Vcle	0.006503	0.007129	[-0.001, 0.014]	0.005944	0.000507	[0.005, 0.006]	Hot side clearance volume
Vswe	0.032035	0.01638	[0.016, 0.048]	0.033492	0.002884	[0.031, 0.036]	Displacer piston swept volume
Phikno	0.000204	0.001274	[-0.001, 0.001]	0.000232	2.3E-05	[0.0, 0.0]	Phase angle
dout	0.583152	0.05658	[0.527, 0.64]	0.583277	0.044271	[0.539, 0.628]	Displacer Cylinder diameter
domat	0.00355	0.006502	[-0.003, 0.01]	0.004978	0.000438	[0.005, 0.005]	Annular gap inner diameter
lc	0.002965	0.005293	[-0.002, 0.008]	0.004036	0.000437	[0.004, 0.004]	Heat exchanger length
dhout	0.001981	0.004158	[-0.002, 0.006]	0.002179	0.000214	[0.002, 0.002]	Housing external diameter
dhin	0.001762	0.003646	[-0.002, 0.005]	0.001588	0.000145	[0.001, 0.002]	Housing internal diameter
dimat	0	0	[0.0, 0.0]	0	0	[0.0, 0.0]	Matrix internal diameter
rl	0.000891	0.003047	[-0.002, 0.004]	0.001265	0.000124	[0.001, 0.001]	Regenerator length
gapout	0.01471	0.010775	[0.004, 0.025]	0.013437	0.001117	[0.012, 0.015]	Annular gap outer diameter
gapin	6.19E-05	0.001	[-0.001, 0.001]	0.000114	1.04E-05	[0.0, 0.0]	Annular gap inner diameter
lh	4.48E-05	0.000553	[-0.001, 0.001]	3.26E-05	3.35E-06	[0.0, 0.0]	Heat exchanger length

Figure 16 GSA Analysis Saltelli

The contribution variables that have high importance in the variability of the response output, are mainly displacer cylinder diameter and cold side clearance volume, following by power piston swept volume. This can be seen in the figure 17.

In the figure 17, we can see a plot where we have sobol sensitivity analysis contribution of each meaningful variable, which verifies the results that were presented earlier.

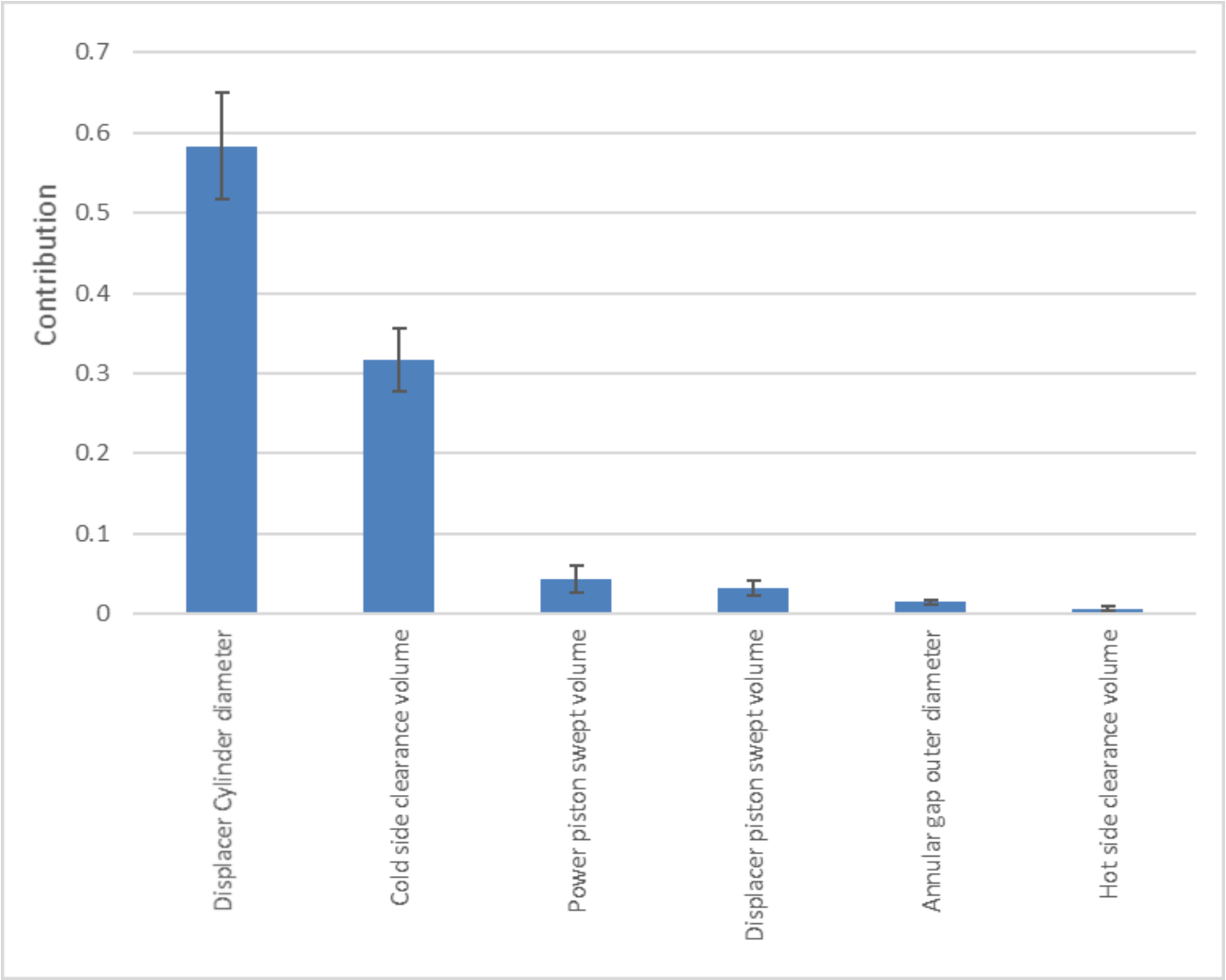

Figure 17 Sobol sensitivity analysis of each meaningful variable

As we can see, both analyses have coherence, first of all with Morris we obtained which variables contribute to obtain the power, and with Saltelli we obtained the percentage contribution of it.

In both analysis the meaningful parameters are displacer cylinder diameter and cold side clearance volume.

This result is logical because both variables are related with the amount of expanding gas that impulses the engine. More surface area will provide to the engine more power.

7. Conclusion

In this thesis, I investigated which is the relation between the different parts of the Stirling engine and the generated thermodynamic power variability.

This thesis showed, through different surrogate models, how the different parts of the engine, represented as inputs, interact and affect in the obtainment of the output, represented as power, in a real Stirling Engine.

This is going to be done using uncertainty quantification (UQ) and sensitivity analysis (SA).

First of all, a set of input variables have been defined. This input variables were the different parts of our Stirling engine.

We have sample them with a very low sample size using Morris to know which variables were meaningful in the contribution of the output power. The contribution variables are mainly displacer cylinder diameter, cold side clearance volume and also power piston swept volume.
On the other hand, there were several variables which do not contribute too much, such as matrix internal diameter and phase angle. These results can be seen in the figure 15.

Then, a UQ analysis have been done, using a larger sample size, of 1000 Sobol samples. The histogram has been drawn in the figure 12.

In order to identify what contributes most, a Sobol indices calculation using Saltelli sampling have been done. This calculation has used 17000 sample and it has been done using a surrogate model.

When the Sobol indices with 17000 surrogate model values is done, the analysis results indicate that the generated power depends mainly of two variables, which are displacer cylinder diameter and cold side clearance volume, as we can see in the figure 17.

To conclude, the result that have been obtained is logical due to the fact that both variables, displacer cylinder diameter and cold side clearance volume, are related with the amount of expanding gas which drives the engine.

So, if the Stirling engine has more surface area, then more generated power will be provided by the engine.

8. References

[1] D. Thombare and S. K. Verma, *Technological development in the Stirling cycle engines*, vol. 12. 2008.

[2] G. Walker, R. Fauvel, R. Gustafson, and J. van Bentham, "Stirling engine heat pumps," *Int. J. Refrig.*, vol. 5, no. 2, pp. 91–97, Mar. 1982.

[3] "Uncertainty quantification." [Online]. Available: https://en.wikipedia.org/wiki/Uncertainty_quantification.

[4] D. Bigoni, "Uncertainty Quantification with Applications to Engineering Problems," 2015.

[5] B. Iooss and P. Lemaître, "A review on global sensitivity analysis methods," *Oper. Res. Comput. Sci. Interfaces Ser.*, vol. 59, no. around 30, pp. 101–122, 2015.

[6] S. Saltelli, A. , Ratto, M. , Andres, T., Campolongo, F. , Cariboni, J. , Gatelli, D. , Saisana, M. and Tarantola, *Global Sensitivity Analysis. The Primer*. 2008.

[7] X. Zhou, H. Lin, and H. Lin, "Global Sensitivity Analysis," in *Encyclopedia of GIS*, Boston, MA: Springer US, 2008, pp. 408–409.

[8] P. Puech and V. Tishkova, "Thermodynamic analysis of a Stirling engine including regenerator dead volume," *Renew. Energy*, vol. 36, no. 2, pp. 872–878, Feb. 2011.

[9] "thermodynamic theory of the ideal stirling engine." [Online]. Available: https://blog.mide.com/thermodynamic-theory-of-the-ideal-stirling-engine.

[10] A. Ross, *Making Stirling Engines*. Ross Experimental, 1993.

[11] S. Alfarawi, R. AL-Dadah, and S. Mahmoud, "Influence of phase angle and dead volume on gamma-type Stirling engine power using CFD simulation," *Energy Convers. Manag.*, vol. 124, pp. 130–140, Sep. 2016.

[12] J. Herman and W. Usher, *SALib: An open-source Python library for Sensitivity Analysis*, vol. 2. 2017.

[13] X. Y. Zhang, M. N. Trame, L. J. Lesko, and S. Schmidt, "Sobol sensitivity analysis: A tool to guide the development and evaluation of systems pharmacology models," *CPT Pharmacometrics Syst. Pharmacol.*, vol. 4, no. 2, pp. 69–79, 2015.

[14] M. D. Morris, "Factorial Sampling Plans for Preliminary Computational Experiments," *Technometrics*, vol. 33, no. 2, pp. 161–174, 1991.

[15] W. U. and others Jon Herman, "basics @ salib.readthedocs.io," 2017. [Online]. Available: https://salib.readthedocs.io/en/latest/basics.html.

[16] "python libraries." [Online]. Available: https://pypi.org/.

[17] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn.*

Res., vol. 12, pp. 2825–2830, 2011.

[18] L. Buitinck *et al.*, *API design for machine learning software: Experiences from the scikit-learn project*. 2013.

[19] W. Chen, R. Jin, and A. Sudjianto, *Analytical Global Sensitivity Analysis and Uncertainty Propagation for Robust Design*, vol. 38. 2006.

[20] H. M. Wainwright, Y. Jung, Q. Zhou, and J. T. Birkholzer, "Making sense of global sensitivity analyses," *Comput. Geosci.*, vol. 65, pp. 84–94, Apr. 2014.

[21] H. Ben Touhami, R. Lardy, V. Barra, and G. Bellocchi, "Screening parameters in the Pasture Simulation model using the Morris method," *Ecol. Modell.*, vol. 266, pp. 42–57, Sep. 2013.

[22] "Sensitivity analysis." [Online]. Available: https://en.wikipedia.org/wiki/Sensitivity_analysis.

[23] "Variance based sensitivity analysis." [Online]. Available: https://en.wikipedia.org/wiki/Variance-based_sensitivity_analysis.

[24] F. Pianosi *et al.*, "Sensitivity analysis of environmental models: A systematic review with practical workflow," *Environ. Model. Softw.*, vol. 79, pp. 214–232, May 2016.

[25] EduPristine, "Sensitivity-analysis," 2018. [Online]. Available: https://www.edupristine.com/blog/all-about-sensitivity-analysis.

[26] "shannon sampling theorem." [Online]. Available: https://www.sciencedirect.com/topics/computer-science/shannon-sampling-theorem.

[27] "Nonuniform sampling." [Online]. Available: https://en.wikipedia.org/wiki/Nonuniform_sampling.

[28] W. U. and others Jon Herman, "salib.readthedocs.io," 2017. [Online]. Available: https://salib.readthedocs.io/en/latest/api.html#method-of-morris.

[29] "surrogates." [Online]. Available: http://sumo.intec.ugent.be/surrogates.

[30] "Surrogate model." [Online]. Available: https://en.wikipedia.org/wiki/Surrogate_model.

[31] Y. University, "Linear regression www.stat.yale.edu," 1997. [Online]. Available: http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm.

[32] M. Peixeiro, "linear regression understanding theory towardsdatascience.com," *2018*. [Online]. Available: https://towardsdatascience.com/linear-regression-understanding-the-theory-7e53ac2831b5.

[33] I. Bhattacharyya, "support vector regression." [Online]. Available: https://medium.com/coinmonks/support-vector-regression-or-svr-8eb3acf6d0ff.

[34] "scikit learn python." [Online]. Available: https://scikit-learn.org/stable/documentation.html.

[35] M. Ebden, *Gaussian Processes for Regression and Classification: A Quick Introduction*. 2015.

[36] C. E. Rasmussen and C. K. I. Williams, “Bayesian Regression and Gaussian processes,” *Gaussian Process. Mach. Learn.*, p. Chapter 2, 2006.

[37] T. Hines, “RBF Documentation,” 2017.

[38] “Radial basis function.” [Online]. Available: https://en.wikipedia.org/wiki/Radial_basis_function.

[39] “Jupyter webpage.” [Online]. Available: https://jupyter.org/.

[40] “stirling engines types @ www.stirlingshop.com.” [Online]. Available: http://www.stirlingshop.com/html/stirlingengines_types.html.

[41] T. Nicr-ni, “Stirling engine Stirling engine,” *Connect*, pp. 1–9, 1816.

[42] I. Urieli, “Author Urieli.” [Online]. Available: https://www.ohio.edu/mechanical-faculty/urieli/index.html.

[43] University Ohio, “Stirling Cycle Machine Analysis,” 2010. [Online]. Available: https://www.ohio.edu/mechanical/stirling/intro.html.

[44] “Standard deviation.” [Online]. Available: https://en.wikipedia.org/wiki/Standard_deviation.

Appendix 1: Matlab code

```
%we have to put the name of the excel file that we want to read it

A = xlsread('input_values2.xlsx');
global new fid
xx = zeros(17:1);

%read the excel file
for i=1:17 %number of rows that we have
    filename = sprintf('input%d.txt',i);
    fileID = fopen(filename,'w');
    B=A(i,:); % read all the elements of the row i
        for j=1:25 %number of columns that we have
            C=A(i,j); %read the element of the row i and column j
            fprintf(fileID, '%f \n', C);

        end

    %now we are going to compute the power of the engine

    % The set of global variables defined are:

    global x
    global mgas
    global pmean
    global vswc vswe
    global rgas
    global alpha
    global vclc vcle
    global vk vr vh
    global tk tr th
    global lr
    global ar
    global dr

    %values obtained from the input file
```

```
vclc=A(i,2);
vswc=A(i,3);
vcle=A(i,4);
vswe=A(i,5);
phase=A(i,6);
dout=A(i,8);
din=A(i,9);
len=A(i,10);
domat=A(i,13);
dimat=A(i,14);
lr=A(i,15);
dhout=A(i,18);
dhin=A(i,19);
lh=A(i,20);
pmean=A(i,22);
tk=A(i,23);
th=A(i,24);
freq=A(i,25);

num = 1;
dcout=dout;

dcin=din;

lc=len;

%inside engine we have this:

alpha = phase * pi/180;

%inside heatex.m we have this:

awgr0 = pi*(dimat + domat)*lr;

amat = num*pi*(domat*domat - dimat*dimat)/4;
ac = num*pi*(dcout*dcout - dcin*dcin)/4;
ah = num*pi*(dhout*dhout - dhin*dhin)/4;
awr = num*pi*(dout*dout - domat*domat)/4;

kwr = 25;
```

```
    cqwr = (kwr*awr)/lr;

    a = pi*(dout*dout - din*din)/4;
    v = a*len;
    awg = pi*dout*len;
    d = dout - din;

    vk = ac*lc;
    vh = ah*lh;
    ar = amat;
    vr = ar*lr;
    awgr = awgr0;
    dr = 4*vr/awgr;

    %for the gas, we have this parameters for air

    rgas = 287.0;

    tr = (th - tk)/log(th/tk);

    c = (((vswe/th)^2 + (vswc/tk)^2 +
2*(vswe/th)*(vswc/tk)*cos(alpha))^0.5)/2;
    s = (vswc/2 + vclc + vk)/tk + vr/tr + (vswe/2 + vcle + vh)/th;
    b = c/s;
    sqrtb = (1 - b^2)^0.5;
    bf = (1 - 1/sqrtb);
    beta = atan(vswe*sin(alpha)/th/(vswe*cos(alpha)/th +
vswc/tk));

    mgas=pmean*s*sqrtb/rgas;

    wc = (pi*vswc*mgas*rgas*sin(beta)*bf/c);
    we = (pi*vswe*mgas*rgas*sin(beta - alpha)*bf/c);
    w = (wc + we);
    power = w*freq;

    x = power;

    format long
    display(x)
    xx(i)=x;
```

```
    fclose(fileID);
    fileID = fopen(['output',num2str(i),'.txt'], 'w');
    fprintf(fileID,'%.15d',x);
    fclose(fileID);

end

xlswrite('output_values2.xlsx',transpose(xx));
```